1. Listing of the claims:

1. (Currently Amended) A system for executing application software on a operating system within a secured run-time environment without affecting an application software resources of a client computer, the system comprising:

an application wrapper, wherein said application wrapper shields the application software resources, whereby said secured run-time environment for executing said application software is created and the application software resources are protected; and

the application wrapper further comprising a privatized virtual file resource created from an operating system file system wherein the application software only accesses the privatized virtual file resource, a privatized virtual registry created from an operating system registry system wherein the application software only accesses the privatized virtual registry, a privatized operating system shared component resource, a privatized application configuration resource and a privatized environmental resources for application variables.

2. (Original) The system of claim 1, wherein privatized virtual file resource further comprising:

intercepting file I/O request generated by one or more processes;

establishing a process ID for the intercepted file I/O request;

comparing process ID to establish operating system process and secured run-time process;

establishing a process ID as operating system process and secured run-time process;

servicing the file I/O request for all process ID established as secured run-time process;

redirecting the file I/O request to operating system service for process ID established as operating system process;

rejecting the file I/O request on secured run-time process resources for process ID established as operating system process;

comparing process ID established as secured run-time process to further establish process resources corresponding to process ID;

establishing corresponding process resources within secured run-time resources; and

rejecting the file I/O request on secured run-time process resources for process ID established as secured run-time process and process ID belongs to other process resources.

3. (Original) The system of claim 1, wherein privatized virtual registry further comprises:

privatizing virtual registry system by intercepting registry I/O request generated by several process;

establishing process ID for the intercepted registry I/O request;

comparing process ID to establish operating system process and secured run-time process;

establishing process ID as operating system process and secured run-time process;

servicing the registry I/O request for all process ID established as secured run-time process;

redirecting the registry I/O request to operating system service for process ID established as operating system process;

rejecting the registry I/O request on secured run-time process resources for process ID established as operating system process;

comparing process ID established as secured run-time process to further establish process resources corresponding to process ID;

establishing corresponding process resources within secured run-time resources; and

rejecting the registry I/O request on secured run-time process resources for process ID established as secured run-time process and process ID belongs to other process resources.

4. (Original) The system of claim 1, wherein privatizing operating system shared component resource further comprising:

searching secured application process for injecting component hooker;

checking the said secured application process to establish whether the process is injected with component hooker;

establishing the said secured application process as new secured application process for said secured application process not injected with component hooker;

injecting component hooker to new secured application process to intercept component process;

repeating component hooker injection for all the new secured application process;

5. (Original) The system of claim 1, wherein privatizing operating system shared component resource further comprising:

Initializing component redirection table to provide component redirecting information;

Registering virtual component required for the secured application;

Adding redirecting information to the said component redirection table for the execution of each selected said secured run-time application;

Removing component redirecting information from the said component redirection table for the termination of each said secured run-time application;

6. (Original) The system of claim 1, wherein privatizing operating system shared component resource further comprising:

intercepting component process function for replacing component search path with secured application resource path;

replacing component search path with private resource path to load the component from the secured application resource path; and

creating new process for the intercepted component with the replaced secured application resource path.

7. (Original) The system of claim 1, wherein privatizing operating system shared component resource further comprising:

intercepting component call for replacing component registry path with the said privatized virtual registry path;

searching component redirection table for the said component redirecting information;

replacing component registry path with the said privatized virtual registry path retrieved from the said component redirection table;

returning the intercepted call to the requested call with the replaced secured application registry path for addressing the component location from the privatized virtual registry system and further the said component is addressed to load from the said privatized virtual file system;

redirecting the said component call as it is to the requested call for the said component call originated from non secured run-time application and for the said component call, which do not have redirecting information in the said component redirection table.

8. (Original) The system of claim 1, wherein privatizing operating system shared component resource further comprising:

intercepting the said RPC message call for replacing component information with privatized virtual component information;

searching component redirecting information from the said component redirection table;

replacing RPC message with the said privatized virtual component information retrieved from the said component redirection table;

returning the intercepted RPC message call to the requested call with the replaced message;

continuing the RPC call to locate the privatized virtual component through SCM;

redirecting the said RPC message call as it is to the requested call for the said component call originated from non secured run-time application and for the said component call, which do not have redirecting information in the said component redirection table.

9. (Original) The system of claim 1, wherein privatized application configuration resource further comprises:

monitoring file I/O request for configuration file to provide separate configuration file;

searching and retrieving configuration file from secured application resources; and

serving application configuration file to requesting process.

10. (Original) The system of claim 1, wherein privatized environmental resources further comprises:

intercepting environment variable request to supply private values to secured application process;

verifying process ID to establish the process ID as operating system process or secured application process;

redirecting the call for process ID established as operating system process;

reading variable data from secured application resource and returning the value to requested process for read variable calls requested by the secured application process;

searching the requesting write variable in secured application resource to find the presence of requesting write variable;

creating variable with variable data within secured application resource and returning the status to requested process for variable do not exist in secured application resource; and updating variable with variable data within secured application resource and returning the status to requested process for variable exist in secured application resource.

11. (Original) The system of claim 1, wherein the application wrapper further comprises selectively allowing the application software to interact operating system resources directly during the said application software executing under the said secured run-time environment.

12. (Original) The system of claim 1, wherein the application wrapper further comprises selectively allowing said application software to interact with other application software resources directly during the said application software executing under the said secured run-time environment.

13. (Original) The system of claim 1, wherein said application wrapper further comprises providing a run-time environment to said application software that is visible to an operating system run-time environment without having said application software run-time resources, whereby said application software resources is simulated to said secured run-time environment during the execution of said application software.

14. (Original) The system of claim 13, further comprising means for protecting the behavior of said application software from other application and said operating system.

15. (Original) The system of claim 13, further comprising means for eliminating said application conflicts from other running application software.

16. (Original) The system of claim 13, further comprising means for executing multiple instance of single said application software.

17. (Original) The system of claim 1, wherein the said application wrapper further comprising maintaining the application software resources away from said operating system resources, whereby said operating system resources is protected from said application software resources.

18. (Original) The system of claim 1, wherein said application wrapper further comprises permitting full access to said application software that requires to access for variation occurs to said application software resources within the said application wrapper.

19. (Original) The system of claim 18 further comprising means for keeping the state of secured run-time environment to said application software.

20. (Original) The system of claim 18, further comprising means for updating said application software resources required by said application software.

21. (Original) The system of claim 1, wherein the said application wrapper monitors the said application run-time request to determine the required said application software resources for execution.

22. (Original) The system of claim 21, further comprising means for receiving said application software resources to execute said application software in the said secured run-time environment.

23. (Original) The system of claim 21, further comprising means for incrementally executing the said application software in the secured run-time environment.

24. (Currently Amended) A method for executing application software on a operating system within a secured run-time environment without affecting an application software resources of a client computer, the client computer ~~compute~~ comprising an application wrapper, wherein said application wrapper shields the said application software resources, whereby a said

secured run-time environment for executing an said application software is created and the said application software resources is protected, the method further comprising:

privatizing virtual file resource created from an operating system file system wherein the application software only accesses the privatized virtual file resource;

privatizing virtual registry created from an operating system registry system wherein the application software only accesses the privatized virtual registry;

privatizing operating system shared component resource;

privatizing application configuration resource; and

privatizing environmental resources for application variables.

25. (Original) The method of claim 24, wherein privatizing the virtual file resource further comprising:

intercepting file I/O request generated by several processes;

establishing process ID for the intercepted file I/O request;

comparing process ID to establish operating system process and secured run-time process;

establishing process ID as operating system process and secured run-time process;

servicing the file I/O request for all process ID established as secured run-time process;

redirecting the file I/O request to operating system service for process ID established as operating system process;

rejecting the file I/O request on secured run-time process resources for process ID established as operating system process;

comparing process ID established as secured run-time process to further establish process resources corresponding to process ID;

establishing corresponding process resources within secured run-time resources; and

rejecting the file I/O request on secured run-time process resources for process ID established as secured run-time process and process ED belongs to other process resources.

26. (Original) The method of claim 24, wherein Privatizing the virtual registry further comprising:

privatizing virtual registry system by intercepting registry I/O request generated by several process;

establishing process ID for the intercepted registry I/O request;

comparing process ID to establish operating system process and secured run-time process;

establishing process ID as operating system process and secured run-time process;

servicing the registry I/O request for all process ID established as secured run-time process;

redirecting the registry I/O request to operating system service for process ID established as operating system process;

rejecting the registry I/O request on secured run-time process resources for process ID established as operating system process;

comparing process ID established as secured run-time process to further establish process resources corresponding to process ID;

establishing corresponding process resources within secured run-time resources; and

rejecting the registry I/O request on secured run-time process resources for process ID established as secured run-time process and process ID belongs to other process resources.

27. (Original) The method of claim 24, wherein privatizing operating system shared component resource further comprising:

intercepting component process function for replacing component search path with secured application resource path;

replacing component search path with private resource path to load the component from the secured application resource path; and

creating new process for the intercepted component with the replaced secured application resource path.

28. (Original) The method of claim 24, wherein privatizing operating system shared component resource further comprising:

searching secured application process for injecting component hooker;

checking the said secured application process to establish whether the process is injected with component hooker;

establishing the said secured application process as new secured application process for said secured application process not injected with component hooker;

injecting component hooker to new secured application process to intercept component process;

repeating component hooker injection for all the new secured application process;

29. (Original) The method of claim 24, wherein privatizing operating system shared component resource further comprising:

Initializing component redirection table to provide component redirecting information;

Registering virtual component required for the secured application;

Adding redirecting information to the said component redirection table for the execution of each selected said secured run-time application;

Removing component redirecting information from the said component redirection table for the termination of each said secured run-time application;

30. (Original) The method of claim 24, wherein privatizing operating system shared component resource further comprising:

intercepting component call for replacing component registry path with the said privatized virtual registry path;

searching component redirection table for the said component redirecting information;

replacing component registry path with the said privatized virtual registry path retrieved from the said component redirection table;

returning the intercepted call to the requested call with the replaced secured application registry path for addressing the component location from the privatized virtual registry system and further the said component is addressed to load from the said privatized virtual file system;

redirecting the said component call as it is to the requested call for the said component call originated from non secured run-time application and for the said component call, which do not have redirecting information in the said component redirection table.

31. (Original) The method of claim 24, wherein privatizing operating system shared component resource further comprising:

intercepting the said RPC message call for replacing component information with privatized virtual component information;

searching component redirecting information from the said component redirection table;

replacing RPC message with the said privatized virtual component information retrieved from the said component redirection table;

returning the intercepted RPC message call to the requested call with the replaced message;

continuing the RPC call to locate the privatized virtual component through SCM;

redirecting the said RPC message call as it is to the requested call for the said component call originated from non secured run-time application and for the said component call, which do not have redirecting information in the said component redirection table.

32. (Original) The method of claim 24, wherein privatizing application configuration resource further comprising:

monitoring file I/O request for configuration file to provide separate configuration file;

searching and retrieving configuration file from secured application resources; and

serving application configuration file to requesting process.

33. (Original) The method of claim 24, wherein privatizing environmental resources for application variables further comprising:

intercepting environment variable request to supply private values to secured application process;

verifying process ID to establish the process ID as operating system process or secured application process;

redirecting the call for process ID established as operating system process;

reading variable data from secured application resource and returning the value to requested process for read variable calls requested by the secured application process;

searching the requesting write variable in secured application resource to find the presence of requesting write variable;

creating variable with variable data within secured application resource and returning the status to requested process for variable do not exist in secured application resource; and

updating variable with variable data within secured application resource and returning the status to requested process for variable exist in secured application resource.

34. (Original) The method of claim 24, wherein selectively allows said application software to interact operating system resources directly during the said application software executing under the said secured run-time environment.

35. (Original) The method of claim 24, wherein selectively allows said application software to interact with other application software resources directly during the said application software executing under the said secured run-time environment.

36. (Original) The method of claim 24, wherein said application wrapper provides an run-time environment to said application software that visible to be an operating system run-time environment without having said application software run-time resources, whereby said application software resources is simulated to said secured run-time environment during the execution of said application software.

37. (Original) The method of claim 36, further comprising protecting the behavior of said application software from other application and said operating system.

38. (Original) The method of claim 36, further comprising eliminating said application conflicts from other running application software.

39. (Original) The method of claim 36, further comprising executing multiple instance of single said application software.

40. (Original) The method of claim 24, wherein the said application wrapper keeps the application software resources away from said operating system resources, whereby said operating system resources is protected from said application software resources.

41. (Original) The method of claim 24, wherein said application wrapper allows full access to said application software that requires to access for variation occurs to said application software resources within the said application wrapper.

42. (Original) The method of claim 41, further comprising a means for keeping the state of secured run-time environment to said application software.

43. (Original) The method of claim 41, further comprising a means for updating said application software resources required by said application software.

44. (Original) The method of claim 24, wherein the said application wrapper monitors the said application run-time request to determine the required said application software resources for execution.

45. (Original) The method of claim 44, further comprising a means for receiving said application software resources to execute said application software in the said secured run-time environment.

46. (Original) The method of claim 44, further comprising a means for incrementally executing the said application software in the secured run-time environment.

2. Amendments to the Specification:

Please replace paragraphs 0004 and 0005 in the published application with the following replacement paragraphs:

Many applications in the past few years have bigger size of application files (element 132 in FIG. 2) and the size has grown dramatically. To reduce the size of the application files 132, the Windows operating environment provides libraries such as COM (common object method), DCOM, IPC to share the modules to Applications. With this environment, the application depends the capability of libraries. A module shared to the applications is said to be a dynamic link library and normally has the extension .DLL. The DLL acts as an application-programming interface (API) that makes Windows work. At the outset, sharing of library modules goes well without a problem. Most applications use only system library and rarely use private libraries. Microsoft windows applications use COM, DCOM, IPC and other libraries either by DLL host or a host process for services (SVCHOST).

Later, with the improvement of windows operating system 100, the library modules come with various versions. In most of the cases, an application probably experienced DLL problems and this may leads the application to behave strangely or no longer loads. This happens due to another program overrides to an older DLL, Visual Basic Extension (VBX) or ActiveX file on their system or may be with an incompatible library version. The application could not run properly due to conflicts by environment settings, registry entries and incompatible library loaded already in the memory.

Please replace paragraphs 0044 and 0045 in the published application with the following replacement paragraphs:

FIG. 9, 10, 14, 15 illustrates more details and steps performed by the privatized virtual component system 146 (shown in FIG. 2) for loading shared component to a private environment. In the present invention, application wrapper 120 includes a component loader, which a module for loading any version of components required for a particular application. For Example: Windows components COM, DCOM, Active X, VBX, Object Linking and Embedding

(OLE) and other application specific components, shared across the operating system 100 to execute several common process. These components are delivered with various features. The components are not stable for all the process. In the preferred embodiment of the present inventions, the privatized virtual component system loads the required version of components for the specific secured application software.

Basically, whenever an application process requests a component then the said component can be searched from the same application process space or from different process space such as Inter Process Communication called IPC, which requires component loading from different process space. The said component calls are processed in different methods in windows operating system. The method includes loading component directly from the file path specified or from the default system directory, loading component based on registry information such as a global unique identifier (GUID) specified, which addresses the component file path through the windows registry, loading component from other process space through service control manager ~~or~~ (SCM) with in-process or out-process technique, which is based on registry information.